Efficient Compositional Modeling for Generating Causal Explanations

P. Pandurang Nayak
RECOM Technologies, Inc.
Artificial Intelligence Research Branch
NASA Ames Research Center
Mail Stop 269-2
Moffett Field, CA 94035-1000

Leo Joskowicz
IBM T.J. Watson Research Center

NASA Ames Research Center

Artificial Intelligence Research Branch

Technical Report FIA-93- 24
August, 1993

Efficient Compositional Modeling for Generating Causal Explanations

P. Pandurang Nayak
Recom Technologies, NASA Ames Research Center
AI Research Branch, MS 269-2
Moffett Field, CA 94035

Leo Joskowicz
IBM T.J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Abstract

Building adequate models that embody the simplifications, abstractions, and approximations that parsimoniously describe the relevant system phenomena for the task at hand is essential for effective problem solving. Compositional modeling is a framework for constructing adequate device models by composing model fragments selected from a model fragment library. This paper presents an implemented polynomial-time model composition algorithm for constructing adequate models that provide parsimonious causal explanations of the functioning of a device. To model important aspects of device function, we introduce expected behaviors, an abstract, causal accounts of *what* a device does. To focus the search for a model and guarantee efficient model construction, we introduce a new approximation relationship between model fragments, called a causal approximation. For efficient model fragment retrieval and model generation, we organize model fragments into various hierarchies. For efficient model validation, we use causal ordering and a new logarithm-based order of magnitude reasoning technique. We have implemented the compositional modeling algorithm and produced adequate models and causal explanations of how a variety of electromechanical devices function, based on a library of 20 component types and 150 model fragments.

1 Introduction

Effective problem solving about complex physical systems requires building models that are both adequate for the task and computationally efficient. Adequate models embody the simplifications, abstractions, and approximations that parsimoniously describe the relevant system phenomena for the task at hand. Overly detailed models are computationally expensive, contain irrelevant information, obscure qualitative differences, and often have unstable solutions or are altogether unsolvable. Overly simplistic models miss or distort key phenomena, ignore relevant interactions, and can lead to erroneous conclusions.

In current practice, most system models are either hand-crafted or are automatically derived assuming the relevant system-wide phenomena have been identified. Hand-crafting models is a difficult, error-prone, and time-consuming activity. It requires skilled and experienced practitioners with a good understanding of what the system does, and how it does it. Recent model-based reasoning systems derive device models from a description of their structure based on so called system-wide or class-wide assumptions. These assumptions determine the simplifications, type of phenomena, and kind of interactions that apply to all the components of the device. For example, in modeling digital circuits, it is customary to have a model for each component that ignores all but its information processing capabilities. While useful, system-wide assumptions limit the modeling scope and fail completely in devices that exhibit many types of phenomena. The goal of automating the model construction process is to overcome these limitations and provide future intelligent programs with an effective modeling tool.

Recent research by Falkenhainer and Forbus [12, 13] proposes *compositional modeling*, an automated modeling framework for constructing adequate device models. Compositional modeling creates models by composing *model fragments* selected from a model fragment library. Model fragments are partial descriptions of components and physical phenomena embodying different assumptions, simplifications, abstractions, and approximations. Model construction is a search process, where the goal is to select an adequate model from the space of possible models defined by the library's model fragments. Compositional modeling is a broad and flexible strategy that can be applied to a variety of tasks. To be effective, it requires a rich, well-organized knowledge base and an effective model search strategy which uses the task and the various available constraints to focus a potentially exponential search process. Falkenhainer and Forbus use compositional modeling within a tutorial setting, and define model adequacy with respect to a set of modeling constraints and a user query. They develop a model composition algorithm that constructs a device model that satisfies the modeling constraints and answers the user query.

In this paper, we address the problem of efficiently generating causal explanations, and the models supporting them, of how a device functions using compositional modeling. Causal explanations have long been known to play a central role in communicating with human users and in automating a vast array of reasoning tasks

[4, 16, 32, 39, 40, 41, 44]. In tutoring, they are used to explain how a device works [4]. In diagnosis, they are used to focus on the components that could have caused a particular symptom [7]. In design, they are used to focus on the mechanisms that can produce the desired behavior [46, 47]. In quantitative analysis, they are used to guide the analysis by providing an overall structure for solving the problem at hand [8]. Models that support causal explanations are necessary to generate causal explanations and are useful to answer queries, run simulations, and make predictions. They can be used as base models that can be fine-tuned and incrementally modified for other tasks.

This paper presents an implemented polynomial-time model composition algorithm for constructing adequate models that provide parsimonious causal explanations of the functioning of a device. The input to the program is the structure of the device (its components, their physical and structural properties, and their interconnections), the *expected behavior* of the device, and the library of model fragments. The output is a causal explanation of the expected behavior and the adequate device model that supports the explanation.

We represent important aspects of device function using expected behaviors, which are abstract, causal descriptions of *what* a device does (but not *how* it does it). Expected behaviors are commonly available either directly from the user, from the description of the problem to be solved, or from the context in which the device operates. For example, expected behaviors are commonly associated with device names: a light bulb transforms electrical energy into light, a vacuum cleaner picks up dust, a disk drive stores and retrieves information. Expected behaviors play a central role in defining model adequacy, in determining relevance of phenomena, and in providing guidance to the model composition algorithm. Without them we cannot discriminate between a myriad of consistent device models: the adequate model for a disk drive, for example, depends on whether it is intended to be used as an information retrieval device, a heater, or a door stop. Adequate device models are required to provide causal explanations for how the expected behavior is achieved.

The structural and behavioral context of the device provide the additional constraints for defining model adequacy and play an important role in model composition. The structural context is defined by the components, their physical and structural properties, and their structural relations, such as their interconnections. In conjunction with the model fragment library, the structural context defines the space of possible device models by identifying the set of applicable model fragments that can be composed to form device models. The behavioral context is determined by the values of the physical parameters used to model the device. It ensures that all significant phenomena are included in the device model, and that all such phenomena are modeled only with applicable approximations.

The focus provided by the task of generating parsimonious causal explanations, and the constraints provided by the structural and behavioral contexts, allowed us to develop an efficient polynomial-time model composition algorithm for generating causal explanations and the models supporting them. We use *causal approximations*

[29, 31], an approximation relationship between model fragments which ensures that the causal relations entailed by a model decrease monotonically as models become simpler. Hence, if a model cannot explain the expected behavior, neither does a simpler model. This property allows us to avoid searching all possible combinations of model fragments, leading to a provably tractable model composition algorithm. We efficiently validate device models against the expected behavior by generating causal explanations using causal ordering [9, 14, 22, 19, 37, 45]. Since device behavior is used to determine significance, we use a new logarithm-based order of magnitude reasoning technique [30] to generate the behavior. We improved the representation of model fragments in the library by organizing them into various hierarchies. This organization provides compact representations, facilitates knowledge-base construction and maintenance, and supports efficient model fragment retrieval and device model generation.

We have implemented the model composition algorithm in Common Lisp and tested it on a variety of electromechanical devices drawn from various sources [2, 25, 38]. For this purpose, we developed a library consisting of 20 types of components including wires, bimetallic strips, springs, and permanent magnets, and 150 types of model fragments, including descriptions of electricity, magnetism, heat, and the kinematics and dynamics of one-dimensional motion. The devices have between 10 and 54 components each, and include different types of temperature gauges, thermostats, relays, and workpiece inspection devices. The number of possible models of each device ranges from about 10^{12} to about 10^{72}. In all cases the program constructs adequate models in 0.5 to 8 minutes on a Texas Instruments Explorer II workstation.

The rest of this paper is organized as follows: Section 2 motivates our approach with an example. Section 3 defines model fragments and describes the model fragment library organization. Section 4 defines model adequacy with respect to the task of generating causal explanations of expected behaviors. Section 5 defines causal approximations and describes the new model composition algorithm. Section 6 presents experimental results from the implementation. Section 7 discusses related work and Section 8 concludes with possible extensions and future work.

2 Example: modeling a temperature gauge

In this section, we motivate the problem and illustrate our approach on a device. Figure 1 shows the schematic of a temperature gauge consisting of a battery, a wire, a bimetallic strip, a pointer, and a thermistor. A thermistor is a semiconductor device; a small increase in its temperature causes a large decrease in its resistance. A bimetallic strip has two strips made of different metals welded together. Temperature changes cause the two strips to expand by different amounts, causing the bimetallic strip to bend. The function of the temperature gauge is to measure the temperature of a liquid in a container by posting its value on a scale. An important aspect of this function is captured by the following expected behavior: the temperature of the

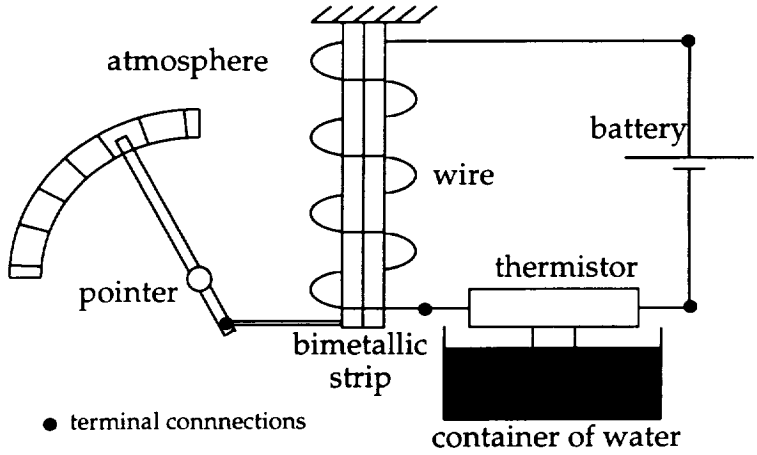

Figure 1: A temperature gauge

thermistor determines the angular position of the pointer.

The temperature gauge achieves its expected behavior as follows: the thermistor senses the water temperature. The thermistor temperature determines the thermistor resistance, which determines the circuit current. This current generates heat in the wire, which determines the temperature of the bimetallic strip. This determines the deflection of the free end of the bimetallic strip, and hence the position of the pointer along the scale.

The main difficulty in modeling this device is accounting for the variety of physical phenomena involved (electrical, thermal, magnetic, kinematic), assessing their relative importance, and selecting among the many different ways in which each component can be modeled and can interact with its neighbors. Within the compositional modeling framework, the different ways of modeling components and component interactions are represented using model fragments and are stored in a model fragment library. Models are constructed by composing an appropriate set of model fragments from this library.

For example, Figure 2 shows some of the model fragments that can be used to model the wire. The wire can be modeled as an electrical conductor, an electromagnet, or an inductor. When it is modeled as an electrical conductor, it can also be modeled either as an ideal conductor or as a resistor. In the latter case, its resistance can be modeled as being constant, or as being dependent upon its temperature. When the wire is modeled as a resistor, it can also be modeled as a thermal resistor, which models the heat generated in the wire due to current flow. Finally, as with most other components, the wire can be modeled using various thermal, mass, and motion models. Similarly, the wire and the bimetallic strip can interact with each other in a number of ways: thermally (heat generated in the wire can cause the bimetallic strip to heat up); magnetically (a magnetic field generated in the wire can cause the bimetallic strip to be magnetized); and kinematically (translation or rotation of the

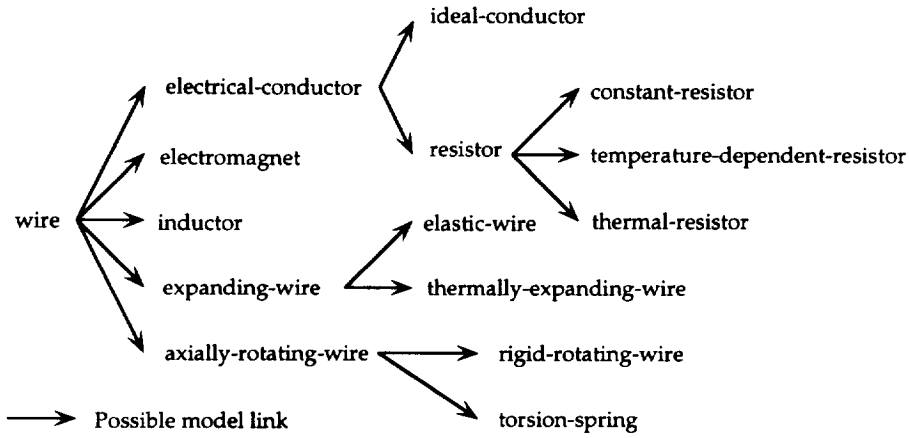

Figure 2: The possible models of a wire.

bimetallic strip causes the same motion in the wire).

The input to our model composition algorithm is the structure of the device (its components and their interconnections), its expected behavior, and a library of model fragments. The algorithm constructs an adequate device model that provides a causal explanation for the expected behavior, and also models all significant phenomena. Such an adequate model can be used to answer a variety of queries related to the functioning of the device. The model composition algorithm constructs adequate models based on the structural and behavioral context of the device, and its expected behavior.

The structure of the temperature gauge, in conjunction with the model fragment library, is instrumental in specifying the space of possible models. First, the components specified in the structure delimit the space, since a device model consists of zero or more models for each of its components. For example, any model of the temperature gauge must include zero or more models of the thermistor, battery, wire, etc. Second, the physical properties of the components delimits the possible ways of modeling them. For example, the wire can be modeled as an electrical conductor only because it is metallic. Third, the relations between the components specify the possible interactions. For example, the wire and the bimetallic strip can interact thermally only because of their relative locations.

The expected behavior determines the set of relevant model fragments that must be part of the model. Since the resulting model must explain the functioning of the temperature gauge, it must include the following component models. The thermistor model must model the effect of its temperature on its resistance. Electrical models are necessary for the wire and the battery, to explain the current flow in the circuit. In addition, the wire must be modeled as a thermal resistor, to explain the generation of heat. Similarly, the thermal properties of the bimetallic strip and the rotation of the pointer must be modeled. The use of these component models is essential for generating the above causal explanation. On the other hand, phenomena such as the

magnetic field generated in the wire, the thermal properties of the battery, and the mass of the thermistor need not be modeled.

Consider now the electrical models selected for the wire and the battery in the above model. From the point of view of generating the above causal explanation, it is sufficient to model the wire as a constant resistance resistor, and the battery as an ideal voltage source, i.e., the temperature dependence of the wire's resistance and the battery's internal resistance need not be modeled. However, suppose that the battery's internal resistance were comparable to the resistance of the wire and the thermistor. To be broadly useful, an adequate models must include all significant phenomena, so that the above model must model the battery's internal resistance. Similarly, if the wire's temperature variation is significant, the temperature dependence of its resistance becomes significant, and must be modeled. To test for significance, we do not need the exact values of the parameters. It is sufficient to compare their orders of magnitude, as engineers do when assessing the relevance of different phenomena.

The resulting model of the temperature gauge satisfies the three important properties of an adequate model. First, it is able to explain the expected behavior, in this case the functioning of the temperature gauge. Second, it includes all significant phenomena. Third, it excludes all irrelevant phenomena, so that it is as simple as possible.

3 Models and model fragments

Device models are formal, abstract descriptions of the device. Different types of device models capture different aspects of the device: structural models describe the parts of the device and how they are put together; behavioral models describe relations between the device's physical attributes, and how these attributes values evolve over time; functional models describe the overall purpose the device and the functional roles of the device's parts that help achieve this purpose.

Our goal is to automatically construct behavioral device models from user-defined structural and (partial) functional device models. Behavioral models are necessary to explain how the device functions, to simulate its behavior over time, and to evaluate its performance. Unlike structural models, which are often directly available from CAD tools, and functional models, which are typically part of the design description, behavioral models must be constructed. Adequate behavioral models must satisfy the constraints imposed by the structure of the device and incorporate the phenomena that describe its function. We view modeling as a cooperative venture between man and machine: the user makes some initial modeling decisions by identifying device components, providing a structural device model, and specifying the functions of interest. The machine then constructs a behavioral model by making a complementary set of modeling decisions.[1]

[1] Henceforth, unless otherwise noted, we will use the term "model" to refer to behavioral models.

We describe models of device behavior by sets of equations that relate sets of device parameters. We focus on time-varying and equilibrium lumped parameter models, which we represent using ordinary differential equations, algebraic equations, and qualitative equations [3]. Different models are described by different sets of equations. Models differ because they give different answers to the following two questions:

- What physical phenomena are to be modeled? Models can differ because they choose to model different physical phenomena.
- How are the chosen phenomena to be modeled? Models can differ because they choose to model the same physical phenomena differently.

Since a device can be modeled in a variety of different ways, it is important to represent the space of possible device models. We represent this space using *model fragments*. A model fragment is a set of equations that partially describes some physical phenomenon. A device model is constructed by composing a set of model fragments, i.e., model fragments are the "building blocks" out of which models are constructed.

Using model fragments to construct device models has three main advantages. First, a set of model fragments is an implicit representation of an exponentially large set of device models: any subset of this set of model fragments can be composed to form a model. This representation, unlike the explicit representation of each model [1], allows broad coverage of phenomena and scales up. Second, model fragments are highly reusable, both in different models of the same device and in models of different devices. Thus, the effort of constructing a library of model fragments can be amortized over their use in a variety of different models. Third, as compared to individual equations, model fragments are better building blocks of models because not all phenomena can be conveniently represented by single equations.

Model fragments relate to each other in meaningful ways. For example, since model fragments can describe different aspects of the same phenomena in different ways, they can be approximations of each other or can be mutually contradictory. Since model fragments can be partial descriptions of phenomena, they might have to be considered jointly to completely describe a particular phenomenon. These relations between model fragments are part of the domain knowledge and are essential for effective model construction. They lead to more compact representations, focus model search, and ease knowledge base building and maintenance.

We describe in detail the model fragments and the model fragment library organization next.

3.1 Model fragments

A model fragment is a set of independent equations that partially describes some physical phenomenon at some level of granularity. Different model fragments can de-

scribe different phenomena, or can be different descriptions of the same phenomenon. For example, the model fragment:

$$\{V_w = i_w R_w\}$$

describes electrical conduction in a wire by modeling the wire as a resistor with voltage V_w, current i_w, and resistance R_w. The model fragment:

$$\{V_w = 0\}$$

describes the same phenomenon for the wire by modeling the wire as an ideal conductor. Finally, the model fragment:

$$\{l_w = l_{w0}(1 + \alpha_w(T_w - T_{w0}))\}$$

describes the temperature dependence of the wire's length, a completely different phenomenon (l_w is the actual length of the wire at temperature T_w, l_{w0} is the original length of the wire at temperature T_{w0}, and α_w is the coefficient of expansion of the wire). Note that, in general, model fragments are only *partial* descriptions of phenomena. For example, the first of the above three model fragments only specifies the relation between the voltage (V_w) and the current (i_w); it does not say anything about the variation of the resistance of the wire. Additional model fragments describing the resistor's resistance are necessary to complete this description.

To determine their activity, model fragments have associated with them *applicability* conditions (also called operating conditions [9, 45] or quantity conditions [14]). Applicability conditions determine when the equations associated with a model fragment can be part of a device model. They are evaluated with respect to the current device state; as the device state changes, the set of active model fragments can change. In this paper, we concentrate on model selection within a single device state, thus assuming that the set of active model fragments is fixed. This set is simply identified by evaluating all applicability conditions against the single device state. Techniques for changing the set of active model fragments with changing device states are discussed in [6, 15, 21].

The equations of a device model are constructed by composing the equations of the model fragments in the model. For algebraic and differential equations, the composition is a simple union of the equations in the model fragments. To combine qualitative equations and expressions with special operators, e.g., direct ($I\pm$) and indirect influences ($\alpha_{Q\pm}$) [14], we use a set of composition rules. Figure 3 shows the model fragments, and the corresponding equations, comprising a model of the temperature gauge of Figure 1. It includes model fragments for the components of the temperature gauge (e.g., the wire `wire-1`) and for the component interactions (e.g., `atmosphere-bms` represents interactions between the atmosphere and the bimetallic strip). The equation *exogenous*(Q) indicates that the value of Q is determined exogenously (it can be viewed as a shorthand for the equation $Q = c$, for some constant c). The equation $M\text{-}(Q_1, Q_2)$ is a qualitative equation indicating a functional dependence between Q_1 on Q_2 and the fact that when Q_2 increases Q_1 decreases [23].

(Constant-temperature atmosphere-1):	$exogenous(T_a)$
(Constant-temperature-model thermistor-1):	$exogenous(T_t)$
(Thermal-thermistor thermistor-1):	$V_t = i_t R_t; \quad M{-}(R_t, T_t)$
(Constant-voltage-source battery-1):	$exogenous(V_v)$
(Constant-resistance wire-1)	$exogenous(R_w)$
(Resistor wire-1):	$V_w = i_w R_w$
(Thermal-resistor wire-1):	$f_w = V_w i_w$
(Equilibrium-thermal-model wire-1):	$f_{wb} = f_w$
(Thermal-bimetallic-strip bms-1):	$x_b = k_2 T_b$
(Equilibrium-thermal-model bms-1):	$f_{ba} = f_{wb}$
(Resistive-thermal-conductor atmosphere-bms):	$f_{ba} = k_3(T_b - T_a)$
(Resistive-thermal-conductor bms-wire):	$f_{wb} = k_4(T_w - T_b)$
Kirchhoff's laws:	$V_v = V_w + V_t; \quad i_v = i_t; \quad i_t = i_w;$ $\theta_p = k_1 x_b$

θ_p: Pointer angle	x_b: Bms deflection
R_w: Wire resistance	R_t: Thermistor resistance
i_t: Thermistor current	V_t: Thermistor voltage
i_w: Wire current	V_w: Wire voltage
i_v: Battery current	V_v: Battery voltage
T_b: Bms temperature	T_a: Atmosphere temperature
T_w: Wire temperature	T_t: Thermistor temperature
f_{ba}: Heat flow (bms to atmosphere)	f_{wb}: Heat flow (wire to bms)
f_w: Heat generated in wire	k_j: Exogenous constants

Figure 3: Model fragments and equations describing the temperature gauge

3.2 The model fragment library and its organization

The model fragment library contains class level descriptions of model fragments, together with relations between these classes. The organization provides compact representations, facilitates knowledge base construction and maintenance, and supports efficient model fragment retrieval and generation of device models. The classes are organized in several different ways:

- into a generalization hierarchy, capturing the subset relation between classes;
- into a possible models hierarchy, capturing alternative ways of modeling components;
- into assumption classes, representing sets of mutually contradictory model fragment classes;
- into an approximation hierarchy, capturing accuracy relations between classes;

- grouped into required assumption classes, indicating model fragment classes that must be used to complete descriptions;

In addition, model fragments are related to each other by articulation axioms, capturing the relationship between terms introduced by different model fragment classes. We discuss each of the above next.

3.2.1 Model fragment classes

Model fragments are represented as classes; a component is modeled by a model fragment by making it an instance of the corresponding class. For example, a wire, `wire-1`, is modeled as a resistor by making it an instance of the `Resistor` class. This modeling choice is represented by the literal `(Resistor wire-1)`, which also denotes the corresponding model fragment. A component can be simultaneously modeled by more than one model fragment simply by making it an instance of each of the corresponding classes. For example, both the electrical and electromagnetic properties of `wire-1` can be modeled by simultaneously making it an instance of `Resistor` and `Electromagnet`.

The model fragment library also contains class level descriptions of the components that can be used to specify device structure. Component classes are similar to model fragment classes, except for their use: component classes are selected by the user to model objects; model fragment classes are automatically selected by the system.

Model fragment classes inherit properties to their instances. These properties comprise the definitions of the phenomena being modeled by the model fragment class. The two most important properties inherited to instances are *attributes* and *equations*. The most common attributes are numerical attributes and terminals. Numerical attributes are the parameters of the phenomena being modeled. Terminals specify the ports through which components can interact with each other by sharing parameters [9].

The equations that a model fragment class inherits to its instances are defined using *equation schemas*. Equation schemas are exactly like equations, except that parameters are replaced by terms such as `(resistance ?object)`. Equation schemas are instantiated for specific instances of the model fragment class by binding the variable `?object` to the instance and replacing the terms by the parameter resulting from evaluating the term.

Figure 4 shows the `Resistor` model fragment class. The `attributes` clause specifies the `resistance` parameter for instances of `Resistor`. The `:range` specification defines the type of this attribute. The `equations` clause specifies the relation between the different parameters.

3.2.2 Generalization hierarchy

Model fragment classes are organized into a generalization hierarchy representing the "subset-of" relation between classes. The generalization hierarchy supports in-

```
(defmodel Resistor
  (attributes
    (resistance
      :range Resistance-parameter
      :documentation "The resistor's resistance"))
  (equations
    (= (voltage-difference ?object)
       (* (resistance ?object)
          (current (electrical-terminal-one ?object)))))
  (generalizations Electrical-conductor)
  (possible-models Constant-resistance
                   Temperature-dependent-resistance
                   Thermal-resistor)
  (assumption-class electrical-conductor-class)
  (approximations Ideal-conductor
                  Ideal-insulator)
  (required-assumption-classes resistance-class))
```

Figure 4: The `Resistor` model fragment class.

heritance, which facilitates knowledge base maintenance and reuse: (a) knowledge represented with a class can be used both by direct instances of the class and by instances of subclasses (specializations) of the class, thereby facilitating its reuse; and (b) since knowledge needs to be represented only with the most general class to which the knowledge is applicable, changes tend to be localized, thereby facilitating knowledge base maintenance.

For example, the `generalizations` clause in Figure 4 states that the `Electrical-conductor` class is a generalization of the `Resistor` class. Hence, any component being modeled as a resistor is also modeled as an electrical conductor.

3.2.3 Possible models hierarchy

We represent the set of model fragments that can describe each component using a *possible models hierarchy*. The possible models of a component or model fragment class are the additional ways of modeling instances of that class. The transitive closure of the possible models of a class is the set of all possible ways of modeling instances of that class.

For example, Figure 2 shows part of the possible models hierarchy rooted at `Wire`. The `possible-models` clause in Figure 4 specifies that instances of `Resistor` can also be modeled as instances of `Constant-resistance`, `Temperature-dependent-resistance`, and `Thermal-resistor`.

(Ideal-conductor wire-1): $V_w = 0$
(Ideal-insulator wire-1): $i_w = 0$
(Resistor wire-1): $V_w = i_w R_w$

Figure 5: Model fragments describing electrical conduction in a wire.

The generalization hierarchy and the possible models hierarchy often overlap. For example, `Resistor` is both a specialization and a possible model of `Electrical-conductor`. However, the two hierarchies are not the same. For example, the `Thermal-thermistor` model fragment class, which models the dependence of a thermistor's resistance on its temperature, is a specialization of the `Thermal-object` model fragment class. However, not all components being modeled as `Thermal-objects` can be modeled as `Thermal-thermistors`: only thermistors can be modeled as `Thermal-thermistors`. Hence, `Thermal-thermistor` is a specialization of `Thermal-object`, but not a possible model of it. Similarly, `Electrical-conductor` is a possible model of `Wire` but not a specialization of `Wire`.

The possible models hierarchy has advantages similar to the generalization hierarchy. First, it leads to compact representations. For example, we only need to specify that instances of `Wire` can be modeled as an `Electrical-conductor`. The additional ways of modeling instances of `Wire` are directly inferred from the hierarchy. Second, it simplifies knowledge base maintenance. For example, adding a model fragment class describing the dependence of resistance on length only requires changing the possible models hierarchy below `Resistor`. Definitions of component and model fragment classes above it, such as `Wire`, need not be modified.

3.2.4 Assumption classes

Different model fragments can be descriptions of different phenomena, or can be different descriptions of the same phenomena. When model fragments describe the same phenomena, they often make contradictory assumptions. We represent this contradictory relation between model fragments by grouping mutually contradictory model fragments into disjoint assumption classes. To avoid inconsistencies, consistent models must include at most one model fragment from each assumption class. Hence, assumption classes can be viewed as a modeling dimension along which some choice needs to be made (including not selecting any model fragments from the assumption class.)

For example Figure 5 shows three mutually contradictory model fragments describing electrical conduction in a wire: the ideal conductor model fragment assumes that the resistance of the conductor is zero, the ideal insulator model fragment assumes that the resistance of the conductor is infinite, and the resistor model fragment assumes that the resistance of the conductor is non-zero and finite.

Note that the contradiction between model fragments in an assumption class cannot, in general, be derived from the equations of the model fragments. For example,

the equations of the ideal conductor model fragment and the ideal insulator model fragment can be simultaneously satisfied when both the current through a conductor and the voltage drop across it are zero. However, these model fragments are mutually exclusive because their underlying assumptions are contradictory. Assumption classes capture this domain-dependent fact.

3.2.5 Approximation hierarchy

Domain experts can often specify that one model fragment is a more *approximate* description of a phenomenon than another model fragment. This indicates that the predictions made by the more accurate model fragment are "closer to reality" than the predictions made by the more approximate model fragment. Typically, more approximate descriptions are simpler to use than more accurate descriptions. Hence, it is desirable to construct models that include all those approximations that do not significantly affect accuracy.

We capture the approximation relation between model fragments by organizing the model fragments within an assumption class into an approximation hierarchy. For example, the `approximations` clause in Figure 4 specifies that `Ideal-conductor` and `Ideal-insulator` are more approximate descriptions of electrical conduction than `Resistor`.

As with the contradictory relation, the approximation relation is a domain-dependent relation that cannot be derived from the equations of model fragments. For example, nothing in the ideal conductor model fragment indicates that it is an approximation of the resistor model fragment; it is a domain fact that must be recorded explicitly by the knowledge engineer.

3.2.6 Required assumption classes

Since model fragments are partial descriptions of phenomena, additional model fragments are sometimes required to complete their description. We represent the set of model fragments that can be used to complete a description by associating with each model fragment a set of *required assumption classes*; the description is completed by including a model fragment from each required assumption class.

For example. the `required-assumption-classes` clause in Figure 4 shows that a component being modeled as a `Resistor` must also be modeled using a model fragment class that specifies `Resistor-class` as its `assumption-class`, viz. `Constant-resistance` or `Temperature-dependent-resistance`.

3.2.7 Articulation axioms

The attributes that a component inherits from different classes are often related to each other. We use a set of rules to ensure that coherent device models incorporate all such relations. These rules are analogous to *articulation axioms* introduced in [18].

For example, `wire-1` inherits the attributes `wire-terminal-one` and `wire-terminal-two`, representing the two ends of the wire. When it is modeled as an `Electrical-conductor` it also inherits the attributes `electrical-terminal-one` and `electrical-terminal-two`, representing the two ends of the electrical conductor. The following rule captures the fact that the two ends of the wire are the two ends of the electrical conductor:

```
(implies
   (and
      (Wire ?object)
      (Electrical-conductor ?object)
      (wire-terminal-one ?object ?term1)
      (wire-terminal-two ?object ?term2))
   (and
      (electrical-terminal-one ?object ?term1)
      (electrical-terminal-two ?object ?term2)))
```

This ensures that components connected to the ends of `wire-1` will be able to electrically interact with it.

4 Adequate models

For any given device, a large number of behavioral models can be assembled from model fragments. The right model is determined by the task for which the model is to be used and the context in which the device operates. For example, to analyze the performance of the temperature gauge in Figure 1 during the final stages of detailed design, it is necessary to include complex non-linear differential equations describing the dependence of the thermistor's resistance on its temperature. However, a simple qualitative current/no current model, explicitly tailored for troubleshooting, is sufficient for determining why the pointer does not move when the water temperature rises.

In this paper, we focus on the task of causally explaining the functioning of a device. Device functioning is represented by the input/output causal dependencies enforced by the device; causal explanations for these dependencies are generated using causal ordering on the equations of the model. This task focus determines the set of relevant phenomena that must be included in an adequate model. The device context is defined by the device's structure and behavior. The structural context is defined by the device's components, their physical and structural properties, and structural relations between them describing how they are put together to form the device. Together with the model fragment library, it defines the space of possible models and provides constraints that guarantee structurally coherent models. The behavioral context is defined by the values, and the variations over time of the values, of the

physical parameters used to model the device. It provides constraints that ensure that all significant phenomena are modeled with applicable approximations.

Regardless of the task and the context, any device model must also be consistent, complete, and parsimonious. Model consistency is enforced by ensuring that models do not contain mutually contradictory fragments. Model completeness is enforced by ensuring that models include complete descriptions of all modeled phenomena, i.e., model fragments from all required assumption classes are included. Model parsimony is enforced by ensuring that only the relevant phenomena are modeled in the simplest possible way. We say that a model is simpler (or more parsimonious) than another if it either models fewer phenomena or does so more approximately.

In summary, an adequate model must satisfy the following criteria:

- **Task adequacy:** the model causally explains the function of the device
- **Contextual adequacy:** the model is consistent with context-dependent structural and behavioral constraints
- **Model consistency:** the model does not contain mutually contradictory model fragment
- **Model completeness:** the model contains a model fragment for each required assumption class
- **Model parsimony:** the model is the simplest model that satisfies the above criteria

Model consistency and model completeness are straightforward. We describe task focus, structural and behavioral contextual adequacy, and model parsimony in detail next.

4.1 Task focus: expected behavior

We define model adequacy with respect to the task of generating parsimonious causal explanations for the functioning of a device. The function of a device is an abstract description of *what* the device does; the causal explanation generated by an adequate model is a description of *how* this function is achieved. The most common functional descriptions are input/output descriptions of device behavior, and correspond to the primary function of the device. Knowledge of device function is commonplace and almost always available either directly from the user, from the description of the problem to be solved, or from the context in which the device operates. For example, the primary function of the temperature gauge in Figure 1 is to measure temperature, so that an adequate model must explain how the thermistor's temperature determines the angular position of the pointer.

We specify device function using causal relations between parameters that must be explained by an adequate device model. We call such required causal relations the *expected behavior* of the device. For example, the expected behavior of the temperature gauge, representing its primary function, is:

```
(causes (temperature thermistor-1)
        (angular-position pointer-1))
```

which states that an adequate device model must explain how the temperature of the thermistor causally determines the angular position of the pointer.

We test if a model satisfies the expected behavior by generating the *causal ordering* of the parameters of the model, using the equations of the model [9, 14, 22, 19, 37, 45]. The causal ordering identifies causal dependencies between the parameters of the model, and hence can be directly used to check if the model explains the expected behavior. Figure 6 shows the causal ordering generated from the equations of the model in Figure 3. It shows that the pointer's angular position (θ_p) is causally dependent on the thermistor's temperature (T_t). Hence, this model explains the expected behavior discussed above.

The expected behavior is a useful characterization of device function. It can also be checked efficiently using the causal ordering algorithm described in [35]. More expressive descriptions of device function, though potentially desirable, are difficult or even impossible to verify. For example, a more expressive description such as "an increase in T_t causes a linear increase in θ_p" is difficult to verify in the presence of non-linear equations and competing influences.

4.2 Structural context

The structural context is defined by the components of the device, their physical and structural properties, and their structural relations. The user models the structure of the device by selecting components from the component types in the model fragment library and by specifying their properties and the structural relations between them. The structural context provides the basis for model construction by defining the space of possible component models and their interactions.

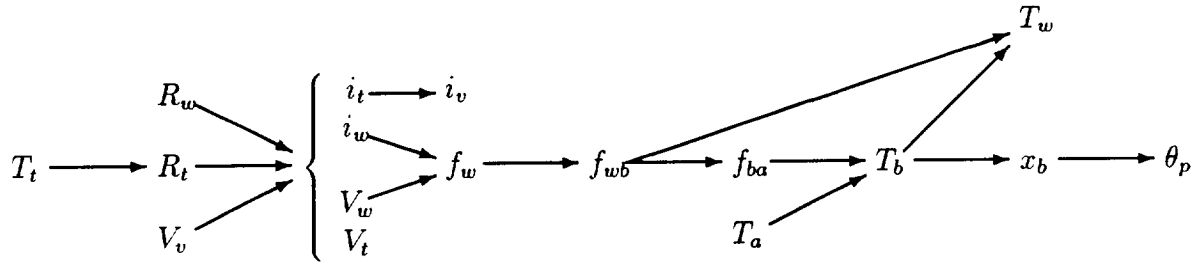

Figure 6: The causal ordering generated from the model in Figure 3. Bracketed parameters are determined simultaneously.

The particular choice of components in the library, their properties, and the possible structural relations reflects the domain of interest and defines the most detailed level of granularity that needs to be considered. For example, in the electromechanical domain, the components of interest include wires, batteries, magnets, and springs. The physical and structural properties include shape, dimension, mass, and material composition. Structural relations, which describe how components are put together, include `connected-to`, indicating that two component terminals are connected to each other [9], `coiled-around`, indicating that a wire is coiled around a component, `meshed` indicating that a pair of gears mesh with each other, and `immersed-in`, indicating that a component is immersed in a fluid. These structural relations determine how components may interact, e.g., the `connected-to` relation supports electrical, thermal, and kinematic interactions between components.

Note that the structural context of a device can change during the operation of the device. It changes as new components are created and old ones are destroyed (e.g., boiling water becoming steam), as the physical properties of components change (e.g., demagnetizing the magnetic strip on a credit card), or as structural relations between components change (e.g., the intermittent contact between the hammer and the dome of an electric bell).

To ensure structurally coherent device models and to model structural context changes, we associate structural constraints with model fragment classes. These constraints state that a component can be modeled by a model fragment class only if it satisfies all the structural constraints associated with that class. We distinguish between two types of constraints: *structural preconditions* and *structural coherence constraints*.

Structural preconditions are necessary constraints on the structural context that must be satisfied if a component is to be modeled by the model fragment class. For example the precondition:

```
(and (composition ?object ?material)
     (metal ?material))
```

in the `Electrical-conductor` model fragment class indicates that a component must be metallic for it to be modeled as an `Electrical-conductor`. Structural preconditions are similar to process preconditions in QP theory [14], except that structural preconditions are necessary conditions, while process preconditions are sufficient conditions.

Structural coherence constraints restrict the model fragment classes that can be used to model a set of structurally related components. For example, the constraint:

```
(implies
   (and (Wire ?object)
        (coiled-around ?object ?core)
        (magnetic-material ?core))
   (Magnet-class ?core))
```

associated with the `Electromagnet` model fragment class specifies that a wire coiled around a core made of magnetic material can be modeled as an electromagnet only if the core is also modeled as a magnet (i.e., by a model fragment class in the `Magnet-class` assumption class). This is because the core amplifies the wire's magnetic field by three or four orders of magnitude, converting the core into a powerful magnet. Without this amplification, the magnetic effect is considered negligible.

4.3 Behavioral context

The behavioral context of a device is defined by the values of the physical parameters used to model the device. The behavioral context changes over time, as the parameter values change over time. For example, the behavioral context of the temperature gauge in Figure 1 includes the values for the current flowing in the circuit, the magnetic field generated by the wire, and the angular position of the pointer.

Because the behavioral context is defined by parameter values, it can only be computed from the model equations *after* a device model has been selected (actual parameter values on a physical prototype are typically unavailable). Since different device models make different modeling assumptions, and hence produce different parameter values, we must ensure that errors introduced by the different modeling assumptions are acceptable. To ensure that models have an acceptable accuracy, we introduce two types of domain-dependent behavioral constraints: *behavioral preconditions*, indicating which approximations are acceptable, and *behavioral coherence constraints*, indicating which phenomena are significant.

Behavioral preconditions are constraints associated with model fragment classes that must be satisfied if a component is to be modeled by that model fragment class. They specify necessary conditions under which an approximate model fragment class can model a component. For example, the behavioral precondition:

```
(< (voltage-difference ?object)
   (voltage-difference-threshold ?object))
```

associated with the `Ideal-conductor` model fragment class indicates that a component can be modeled as an ideal conduct, rather than more accurately as a resistor, only if the voltage drop across it is less than some threshold. Behavioral preconditions are similar to process quantity conditions in QP Theory [14]. However, behavioral preconditions are modeling constraints used to decide which model fragment classes in an assumption class can model a component, while quantity conditions control the activity of a process, and are about the physics of the situation.

Behavioral coherence constraints, like structural coherence constraints, constrain the model fragment classes that can model a set of related components. They ensure that device models include all significant phenomena. For example, the constraint:

```
(implies
  (>= (* (voltage-difference ?object)
         (current (electrical-terminal-one ?object)))
      (electrical-power-threshold ?object))
  (Thermal-resistor-class ?object))
```

in the `Resistor` model fragment class specifies that when the power dissipation of a component modeled as a resistor exceeds a prespecified threshold, this power dissipation is deemed significant and must be explicitly modeled by adding a model fragment class from the `Thermal-resistor-class` assumption class to the wire model.

Parameter thresholds, such as the electrical power dissipation threshold in the previous example, play a central role in determining the significance of phenomena, and the applicability of approximations. Depending on the situation, thresholds can either be preset or computed dynamically. Preset thresholds can be derived from physics, such as the Reynolds number in fluid dynamics indicating when laminar flow becomes turbulent, or can be set by an engineer depending on the application. For example, voltage differences of up to 10 volts can be considered insignificant in a power distribution system, while voltage differences of only up to .01 volts can be considered insignificant in an electronic circuit. Thresholds can also be set dynamically, based on knowledge of acceptable error tolerances on parameters. The error tolerances can be propagated, via a set of rules or through the model equations, to set other thresholds [27, 36].

4.4 Model parsimony

Model parsimony guarantees that only relevant phenomena are modeled in the simplest possible way. To compare models, we establish a partial order between models based on the intuitions that a model is simpler if it models fewer phenomena, and that approximate descriptions are simpler than more accurate ones. Hence, we say that model M_2 is *simpler* than model M_1 if and only if for each model fragment $m_2 \in M_2$ either (a) $m_2 \in M_1$; or (b) there is a model fragment $m_1 \in M_1$ such that m_2 is an approximation of m_1.

For example, a model simpler than the temperature gauge model in Figure 3 ignores the wire's heating properties by removing the model fragment `(Thermal-resistor wire-1)`. A more complex model takes into account the thermal properties of the wire's resistance by replacing the model fragment `(Constant-resistance wire-1)` with the more accurate model fragment `(Temperature-dependent-resistor wire-1)`. An incomparable model results from both removing `(Thermal-resistor wire-1)` and replacing `(Constant-resistance wire-1)` by `(Temperature-dependent-resistor wire-1)`.

Note that the definition of model simplicity does not necessarily guarantee that simpler models will be more efficient to simulate or will produce simpler causal explanations than more complex ones. However, it is a good heuristic for identifying

computationally efficient models and for generating parsimonious causal explanations because it follows common engineering practice which simplifies models by disregarding irrelevant phenomena and by using all applicable approximations.

5 Finding adequate models

Given a device description, its expected behavior, and a library of model fragments, we construct an adequate model by composing a set of model fragments describing the device's components and their interactions. Despite the focus and constraints imposed by the causal explanation task, finding an adequate model is intractable—in the worst case, we must consider exponentially many combinations of model fragments before finding an adequate model. Nayak [29, 28, 31] formally shows that the problem is NP-hard and identifies three sources of intractability: (a) deciding which assumption classes to include in the model; (b) deciding which model fragments to choose from selected assumption classes; and (c) satisfying the structural and behavioral coherence constraints.

Deciding which assumption classes to include in the model does not appear to be intractable in practice, so we do not address it. We address the second source of intractability by requiring that all the approximation relations between model fragments in the model fragment library (Section 3.2.5) must be *causal approximations* [29, 28, 31]. An approximation relation between two model fragments is said to be a causal approximation if the more approximate model fragment contains equations with fewer parameters than the more accurate model fragment, so that the more approximate model fragment explains less about the phenomenon. Since simpler models contain more approximations, it follows that when all approximations are causal approximations, the causal relations entailed by a model decrease monotonically as models become simpler. Hence, if a model does not explain the expected behavior, neither does any simpler model. This property allows us to focus on a small number of model fragment combinations.

We address the third source of intractability by restricting the expressivity of coherence constraints. In particular, we require that the coherence constraints be like horn clauses, except that the consequent requires that a component be modeled using any model fragment class from an assumption class, rather than a specific model fragment class, i.e., the consequent is the disjunction of the model fragments in an assumption class. We also assume that if a model satisfies all the coherence constraints, then any consistent simpler model that uses model fragments from the same assumption classes also satisfies the constraints. These restrictions ensure that a model can be simplified efficiently: if a model satisfies the coherence constraints then a simpler model produced by replacing a model fragment by an approximation is guaranteed to satisfy the coherence constraints. Model simplification is a key step in our model composition algorithm. These restrictions have not proved to be significant, since our definition of model adequacy is primarily driven by the expected behavior,

not the coherence constraints. For a complete analysis, see [28].

We find an adequate model by first identifying an initial model that satisfies all the model adequacy criteria with the exception of model parsimony. We then simplify this initial model until none of its immediate simplifications explains the expected behavior. The use of causal approximations ensures that the resulting model is adequate, while the restriction on the coherence constraints ensures that simplification is efficient.

We construct the initial model in five steps. First, we augment the initial device description to include all expected behavior parameters. Second, we augment the device model using the structural coherence constraints and a set of *component interaction constraints*. These constraints ensure that every component interaction that can possibly take place is included in the model. Third, we generate the device behavior based on the current choice of model fragments using a logarithm-based order of magnitude reasoning technique. Fourth, we augment the device model using the behavioral coherence constraints and examine the resulting device model. If the model does not explain the expected behavior, we further augment it in the fifth step by choosing alternate model fragments. We repeat these steps until an initial model is found. We then simplify the initial model by attempting to replace model fragments with one of their approximations or by dropping them altogether.

Our model composition algorithm has three main advantages over earlier approaches [13, 31]. First, it avoids producing the most accurate initial model, significantly speeding up model simplification. Second, it uses an efficient order of magnitude reasoning technique that is at the right level of detail for the task, striking a balance between qualitative and quantitative techniques. Third, unlike [13], it has a guaranteed polynomial time complexity.

The remaining of this section briefly summarizes causal approximations, order of magnitude reasoning, introduces component interactions constraints, and describes the model composition algorithm in detail.

5.1 Causal approximations

To make the model fragment selection tractable, we require that all approximations in the model fragment library be *causal approximations*. The key idea underlying the definition of causal approximations is that approximate descriptions often use fewer parameters than more accurate descriptions. As a result, more approximate descriptions explain less about a phenomenon than more accurate descriptions. Using this observation, we define causal approximations as follows: every equation in the more approximate model fragment must have a corresponding equation in the more accurate model fragment which uses a superset of parameters. Hence, when all approximations are causal approximations, simpler models contain fewer parameters and explain less than more complex models, i.e., the causal relations entailed by a model decrease monotonically as models become simpler. A rigorous analysis of the properties of causal approximations is found in [28, 31].

A central property of monotonically decreasing causal relations is that if a model does not explain the expected behavior, no simpler model can. Hence, an adequate model can be identified in two steps: (a) identify a model that explains the expected behavior; and (b) simplify the model as much as possible, until none of the immediate simplifications of the model can explain the expected behavior. The monotonicity of the causal relations assures us that the resulting model is an adequate model. Modeling the different aspects of the physical world with causal approximations is not a limitation. It is both natural and commonplace in physics and engineering textbooks.

To illustrate the monotonicity of causal relations, consider the model fragments in Figure 7, with `(Constant-resistance wire-1)` an approximation of `(Temperature-dependent-resistance wire-1)`. Note that this approximation is also a causal approximation, since the parameter in $exogenous(R_w)$ is also used in $R_w = R_{w0} + \alpha_w(T_w - T_{w0})$. The model in Figure 3 uses the model fragment `(Constant-resistance wire-1)`, and Figure 6 shows the corresponding causal ordering. Consider replacing `(Constant-resistance wire-1)` in this model by `(Temperature-dependent-resistance wire-1)`. Figure 8 shows the causal ordering generated from this more complex model. The causal relations entailed by this causal ordering are a superset of the causal relations entailed by the causal ordering in Figure 6: there are three more parameters, T_{w0}, R_{w0} and α_w, causally related to R_w and a new causal link from T_w to R_w in the new model. Many additional examples of causal approximations, such as elastic collisions, frictionless motion, and ideal gas laws are described in [28, 31].

5.2 Order of magnitude reasoning

Generating the behavior of the device during model construction is necessary to evaluate the behavioral constraints and to determine the significance of different physical phenomena. The equations in a device model can seldom be solved in closed form. Hence, the most widely used techniques for solving a set of lumped-parameter equations are numerical and qualitative methods. Numerical methods require exact values for exogenous parameters, which are not always available, and can be unstable, inefficient, or converge to the wrong solution. Qualitative methods, which consider only parameter signs [3, 44], lack the discriminatory power to estimate the significance

`(Constant-resistance wire-1)`:	$exogenous(R_w)$
`(Temperature-dependent-resistance wire-1)`:	$R_w = R_{w0} + \alpha_w(T_w - T_{w0})$
	$exogenous(R_{w0})$
	$exogenous(\alpha_w)$
	$exogenous(T_{w0})$

Figure 7: Model fragments for the wire's resistance.

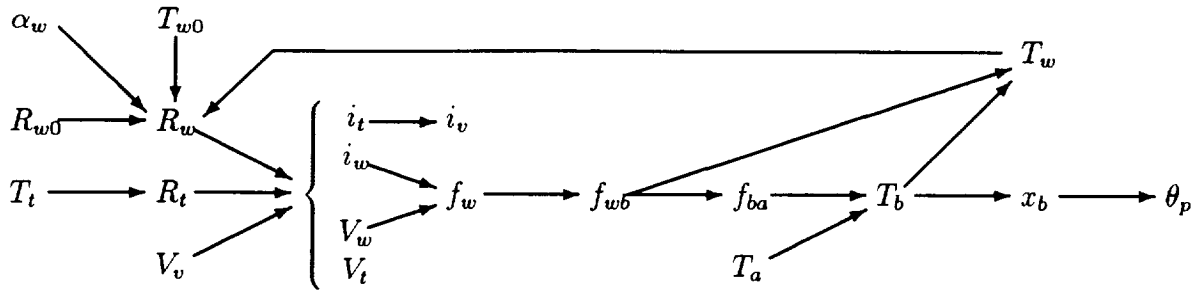

Figure 8: The causal ordering generated from the model resulting from replacing (`Constant-resistance wire-1`) by (`Temperature-dependent-resistance wire-1`) in Figure 3

.

of phenomena and can lead to ambiguity. Order of magnitude reasoning strikes a balance between these two extremes.

We use the order of magnitude reasoning technique embodied in a program called NAPIER[30]. NAPIER defines the order of magnitude of a quantity on a logarithmic scale and uses a set of rules to propagate orders of magnitude through equations. It handles non-linear simultaneous equations and uses approximation techniques to make the computation tractable and efficient. We use the parameter orders of magnitude computed by NAPIER to determine the relative orders of magnitude of parameters, which allows us to determine the phenomena that are significant and worth modeling. In particular, we use the computed orders of magnitude to evaluate behavioral preconditions and coherence constraints. These determine which model fragments should or should not be part of the device model.

Following is a brief description on NAPIER's operation (see [30] for a detailed account). NAPIER defines the order of magnitude of a parameter q (denoted $om(q)$) as an interval on a logarithmic scale:

$$om(q) = \lfloor \log_b |q| \rfloor \tag{1}$$

where the base, b, of the logarithm is chosen to be the smallest number that can be considered to be "much larger" than 1 (we use $b = 10$ in this paper). NAPIER uses a set of rules to propagate orders of magnitude through equations. The orders of magnitude are propagated through the equations in two steps. First, a set of rules convert each equation into a disjunction of sets of linear inequalities. For example, the equation

$$q_3 = q_1 * q_2$$

is converted into the set of linear inequalities (in this case, a single disjunct):

$$\{om(q_1) + om(q_2) \leq om(q_3), om(q_3) \leq om(q_1) + om(q_2) + 1\}$$

The result of this conversion is a disjunctive logic program. Second, the program solves this disjunctive logic program using linear programming and backtracking.

Since solving the disjunctive logic program is, in general, NP-hard, NAPIER uses a heuristic method based on causal ordering. This heuristic method is fast and does not appear to lose accuracy in practice. For example, NAPIER solves a set of 163 equations in only 21 seconds on a Texas Instruments Explorer II workstation, with no loss of accuracy [30].

To illustrate how orders of magnitude are used in model selection, consider modeling a wire through which current is flowing. Suppose that NAPIER has predicted that the order of magnitude of the current through the wire is -1 (several deciamperes), and that of the voltage across the wire is 0 (several volts). To determine if the heat generated in the wire is significant and must be modeled, recall the behavioral coherence constraint in Section 4.3:

```
(implies
  (>= (* (voltage-difference ?object)
         (current (electrical-terminal-one ?object)))
      (electrical-power-threshold ?object))
  (Thermal-resistor ?object))
```

where the order of magnitude of the electrical power threshold is -1 (several deciwatts). Following the previous order of magnitude rule, the order of magnitude of the product of the current and the voltage is between -1 and 0, which is greater than or equal to the electrical power threshold. This indicates that the heat generated by the wire is significant and should be modeled as a `Thermal-resistor`.

5.3 Component interaction heuristic

To focus the search for initial adequate models, we introduce the *component interaction heuristic*. Components can only interact with each other if the components are related by the structural relations that support the interaction and the component models are compatible with the interaction. For example, two wires can interact electrically only if they are connected to each other and if they are both modeled as electrical conductors. The heuristic requires that if a set of components are related by one or more structural relations that support an interaction, and if one of the component models is compatible with this interaction, then the remaining component models must be augmented to be compatible with this interaction. This allows the components in the set to interact with each other via that interaction. Note that if none of the component models is compatible with the interaction, then no augmentations are necessary.

While we require the initial model to satisfy the additional constraints imposed by this heuristic, the final adequate model need not satisfy them. The interactions might eventually be discarded because they may be too weak to be worth modeling or may not be relevant in explaining the expected behavior.

We implement the component interaction heuristic with a set of constraints. Each constraint specializes the heuristic for a particular interaction and structural relations. For example, the constraint:

```
(implies
   (and (terminals ?object ?term1)
        (voltage-terminal ?term1)
        (connected-to ?term1 ?term2)
        (terminal-of ?term2 ?comp2))
   (electrical-component ?comp2))
```

in the `electrical-component` model fragment class says that if a component is being modeled as an `electrical-component`, and one of the component's voltage terminals is connected to a terminal of another component, then the other component must also be modeled as an `electrical-component`. This allows the two components to interact by sharing voltages at the connected terminals. We require the initial model to satisfy all the component interaction constraints.

5.4 Model composition algorithm

Figure 9 shows the algorithm for finding a device model that explains the expected behavior of a device. The inputs to the algorithm are:

- the structure of the device: its components, their physical and structural properties, and the structural interconnections between them;
- the expected behavior of the device;
- orders of magnitudes of thresholds and exogenous parameters; and
- an optional set of model fragment classes preselected for each component, corresponding to modeling decisions made by the user.
- the library of model fragments organized as described in Section 3.

The algorithm starts by identifying all possible component interaction paths with a set of rules (step 0 in the flowchart). The paths are modeled with additional device components and are treated exactly like the original components by the remaining steps in the algorithm.

The rest of the algorithm has six steps. All but step three (generate behavior) and step six (simplify model) can modify the device model by adding one or more model fragments to it. When a model fragment is added to the device model, the most accurate model fragment from every assumption class required by the model fragment is also added to the model. Thus, at the end of every step, the device model always contains a model fragment from all required assumption classes.

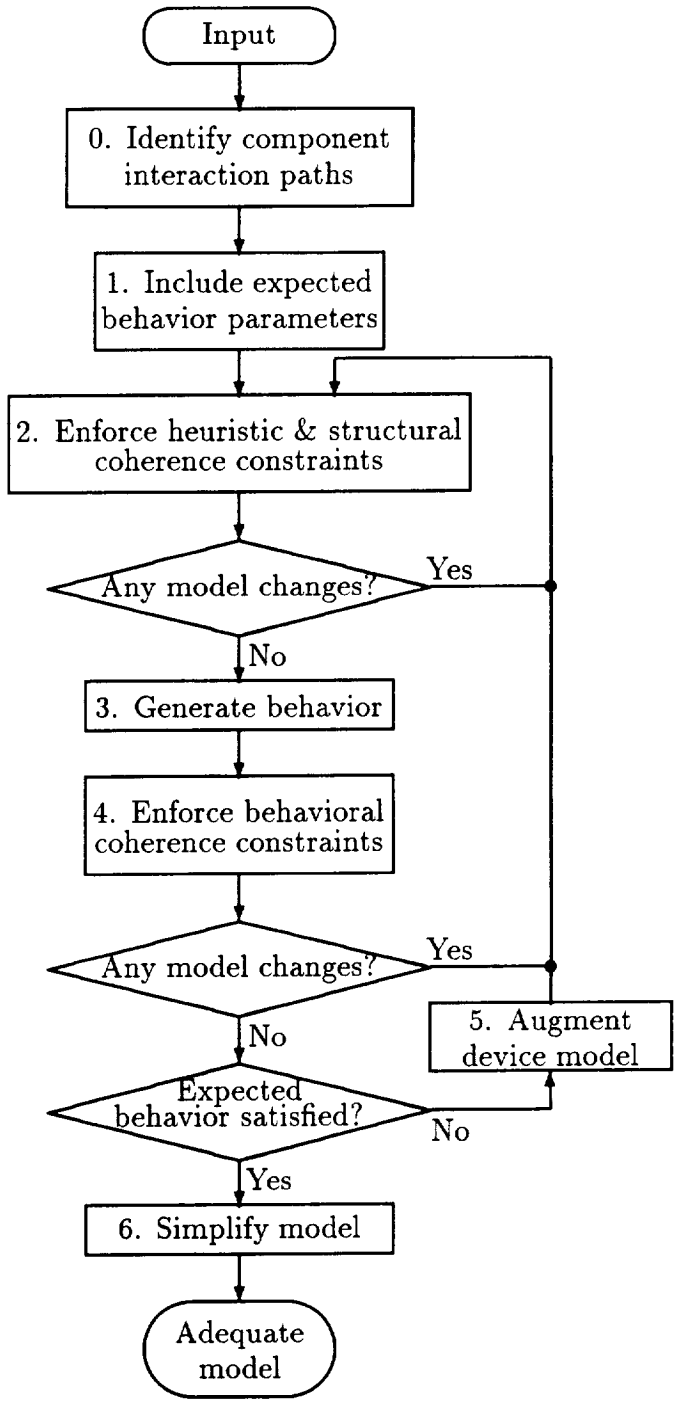

Figure 9: Model composition algorithm

In the first step, the algorithm checks if the input device model contains all expected behavior parameters. If a component parameter is missing, the algorithm searches the possible models hierarchy rooted at that component for a model frag-

ment class that provides the required parameter and whose structural preconditions are all satisfied. The resulting model fragment class is added to the component's model. When several model fragment classes satisfy the above condition, we prefer (a) more general model fragment classes over more specific ones to ensure that minimal modeling commitments are made; (b) only most accurate model fragment classes, i.e., only model fragment classes that are not approximations of any other model fragment class (this is necessary because we cannot determine the applicability of approximations by evaluating behavioral preconditions since the behavior has not been generated).

In the second step, the program checks the structural and component interaction constraints of each component model. An unsatisfied constraint indicates that the component model does not include a required model fragment. The algorithm then searches the possible models of that component for a model fragment class that, when added to the component model, would satisfy the violated constraint. As before, the algorithm only considers most accurate model fragment classes whose structural preconditions are satisfied. The component model is augmented with the resulting model fragment class. This step is repeated until all structural and component interaction constraints are satisfied.

In the third step, the algorithm uses the current device model to generate the device behavior using NAPIER. NAPIER computes the orders of magnitudes of all parameters from the exogenous parameter values provided as input by the user.

In the fourth step, the algorithm uses the behavior generated above to enforce all the behavioral coherence constraints. This step is exactly analogous to the way structural coherence constraints were enforced in the second step. If there are any changes to the device model, the algorithm loops back to the second step, to ensure that all the structural and heuristic coherence constraints continue to be satisfied. The algorithm loops through steps two, three, and four, until all structural, behavioral, and component interaction constraints are satisfied.

Once all these constraints are satisfied, the algorithm checks to see if the resulting model can explain the expected behavior. This check is done by generating the causal ordering of the model parameters using the efficient causal ordering algorithm presented in [35]. Using simple graph traversal, the algorithm then checks whether the causal ordering can explain the causal relations required by the expected behavior. If the expected behavior is explained by the causal ordering, the algorithm proceeds to the sixth step. Otherwise, it proceeds to the fifth step.

In the fifth step, the algorithm augments the device model with the alternate model fragment classes identified in the earlier steps. These alternatives are either model fragment classes that are more specific than those chosen earlier, or other most general model fragment classes that could have been, but were not, selected. The algorithm then loops back to the second step, until a model that explains the expected behavior is found. If all possible ways of augmenting the device model are exhausted, then the algorithm reports that the expected behavior cannot be explained. This is justified because the component interaction heuristic guarantees

that the component models used in the final device model cannot interact with any other components, and hence no augmentation of the device model can lead to a model that explains the expected behavior.

The sixth and last step is model simplification. The initial model produced by the previous steps can be more complex than necessary for one of three reasons: (a) for each required assumption class, we added in the most accurate model fragment, even though a more approximate model fragment might do; (b) model fragments added to satisfy the component interaction constraints are not strictly necessary; and (c) unnecessary model fragments may have been added in step five.

We simplify the initial device model by applying one of the following two simplification operators: (a) replace a model fragment by one of its immediate approximations; and (b) remove a model fragment. The algorithm simplifies the model by repeatedly applying the simplification operators while ensuring that all structural and behavioral constraints are satisfied and that the expected behavior can be explained. When all immediate simplifications of a model are unable to explain the expected behavior, the program terminates, and returns that model as an adequate model. It also generates a causal explanation for the expected behavior using causal ordering as described earlier. Note that the application of different sequences of simplification operators can result in different models. However, the different models differ in features deemed to be insignificant by the behavioral constraints and thresholds, and hence the program returns the first adequate model it finds.

The number of device models constructed by the above algorithm is linear in the number of model fragments. In particular, steps 1–5 only modify the model by adding zero or more model fragments. Since only consistent models are constructed, these steps can add at most one model fragment from each assumption class. Hence, the number of models constructed by this part of the algorithm is bounded by the number of assumption classes. Since the assumption classes are disjoint, it follows that the number of assumption classes, and hence the number of models constructed in steps 1–5, is bounded by the number of model fragments. Step 6 simplifies the initial model by either dropping a model fragment or by replacing a model fragment by one of its approximations. In either case, once a model fragment has been removed from the model, it is never reconsidered. Hence, the number of models constructed by step 6 is bounded by the total number of model fragments. Hence, the number of device models constructed by the above algorithm is linear in the number of model fragments.

5.5 Example: modeling the temperature gauge

We now illustrate the above algorithm on the temperature gauge in Figure 1. Part of the input, describing the components and their component classes, is shown in the first two columns of Table 1. The last four rows show the components added by the algorithm to model component interaction paths in step zero. For example, `bms-wire` is a possible interaction path between `bms-1` and `wire-1`, corresponding to `wire-1`

being coiled-around bms-1.

In the first step, the expected behavior of the temperature gauge:

```
(causes (temperature thermistor-1)
        (angular-position pointer-1))
```

requires a temperature parameter for the thermistor and an angular-position parameter for the pointer. The former can be achieved by modeling the thermistor either as a Thermal-object or as a Thermal-thermistor. The algorithm uses Thermal-object since it is more general. Similarly, the pointer is modeled as a Rotating-object. Finally, Dynamic-thermal-model is added to the thermistor model, since it is the most accurate model fragment class of the assumption class required by Thermal-object.[2] The resulting model is shown in the third column of Table 1.

In the second step, because the pointer is a Rotating-object that is connected to the free end of the bimetallic strip, a component interaction constraint requires a kinematic interaction between pointer-1 and bms-1. This constraint can be satisfied by modeling the bimetallic strip as a Thermal-bimetallic-strip, which models the deflection of the free end of the bimetallic strip as a function of its temperature. As a consequence of this modeling decision, another component interaction constraint requires a thermal interaction between bms-1 and both wire-1 (via bms-wire) and atmosphere-1 (via atmosphere-bms), so that wire-1 and atmosphere-1 are modeled as Thermal-objects and bms-wire and atmosphere-bms are modeled as

[2]We will not mention these required assumption class augmentations in the rest of the example.

Component	Component classes	After step 1	After step 2
thermistor-1	Thermistor	Thermal-object Dynamic-thermal-model	Thermal-object Dynamic-thermal-model
pointer-1	Pointer	Rotating-object	Rotating-object Thermal-object Dynamic-thermal-model
bms-1	Bimetallic-strip		Thermal-bimetallic-strip Dynamic-thermal-model
wire-1	Wire		Thermal-object Dynamic-thermal-model
battery-1	Battery		Thermal-object Dynamic-thermal-model
atmosphere-1	Atmosphere		Thermal-object Dynamic-thermal-model
bms-wire	Coil-structure		Resistive-thermal-conductor
atmosphere-pointer	Immersion-structure		Resistive-thermal-conductor
atmosphere-bms	Immersion-structure		Resistive-thermal-conductor
atmosphere-battery	Immersion-structure		Resistive-thermal-conductor

Table 1: Components and their initial models.

Resistive-thermal-conductors. Modeling atmosphere-1 as a Thermal-object requires that all objects immersed in it must also have thermal models. The resulting device model is shown in the fourth column of Table 1.

In the third step, the algorithm uses NAPIER to generate the behavior, and then proceeds to check the behavioral coherence constraints in the fourth step. None of these constraints are currently violated, so we now have a device model that satisfies the structural, behavioral, and component interaction constraints. However, the expected behavior is not satisfied, so the algorithm proceeds to the fifth step.

Recall that an alternate way of providing the temperature parameter to thermistor-1 was to model it as a Thermal-thermistor. Hence, the fifth step augments the thermistor model with this model fragment class, and returns to the second step. Since Thermal-thermistor is an electrical model, a component interaction constraint requires an electrical interaction between the thermistor and the wire and battery. This is achieved by modeling the wire as a Resistor, and the battery as a Voltage-source-with-resistance. The resulting model is used to generate the behavior, which includes calculating the wire's voltage and current. The behavioral coherence constraint:

```
(implies
  (>= (* (voltage-difference ?object)
         (current (electrical-terminal-one ?object)))
      (electrical-power-threshold ?object))
  (Thermal-resistor ?object))
```

requires that the wire must be modeled as a Thermal-resistor, since the product of the wire's voltage and current exceeds its electrical-power-threshold. With this augmentation all the structural, behavioral, and heuristic constraints are satisfied, and the expected behavior explained. The resulting initial model is shown in the second column of Table 2.

In the sixth step the initial model is simplified by approximating and dropping model fragments. For example, in the initial model the battery is a Voltage-source-with-resistance. However, the internal resistance of the battery is very small, so that the approximation Constant-voltage-source is applicable. Similarly, wire-1's resistance can be assumed to be constant, rather than temperature dependent. The simplification process also determines that the thermal properties of pointer-1 and battery-1 are irrelevant, and hence it drops the corresponding model fragment classes from the model. The adequate model, resulting from this simplification process, is shown in the third column of Table 2.[3]

[3]This column includes model fragments that only provide parameters, but do not directly introduce equations into the model (e.g., (Rotating-object pointer-1)). Hence, the model fragments in this column are a superset of the model fragments listed in Figure 3.

Component	Initial Model	Adequate Model
thermistor-1	Thermal-object Dynamic-thermal-model Thermal-thermistor	Thermal-object Constant-temperature-model Thermal-thermistor
pointer-1	Rotating-object Thermal-object Dynamic-thermal-model	Rotating-object
bms-1	Thermal-bimetallic-strip Dynamic-thermal-model	Thermal-bimetallic-strip Equilibrium-thermal-model
wire-1	Thermal-object Dynamic-thermal-model Electrical-conductor Resistor Temperature-dependent-resistance Thermal-resistor	Thermal-object Equilibrium-thermal-model Electrical-conductor Resistor Constant-resistance Thermal-resistor
battery-1	Thermal-object Dynamic-thermal-model Voltage-source Voltage-source-with-resistance	 Voltage-source Constant-voltage-source
atmosphere-1	Thermal-object Dynamic-thermal-model	Thermal-object Constant-temperature-model
bms-wire	Resistive-thermal-conductor	Resistive-thermal-conductor
atmosphere-pointer	Resistive-thermal-conductor	
atmosphere-bms	Resistive-thermal-conductor	Resistive-thermal-conductor
atmosphere-battery	Resistive-thermal-conductor	

Table 2: The initial and adequate model.

6 Implementation and results

We have implemented the model composition algorithm in Common Lisp and tested it on a variety of electromechanical devices. This section presents the results of the implementation.

We constructed a library of 20 different types of components, such as wires, bimetallic strips, springs, and permanent magnets. The library consists of approximately 150 different types of model fragment classes including descriptions of electricity, magnetism, heat, elasticity, and the kinematics and dynamics of fixed-axes rotation and translation. Each component class has an average of 30 model fragment classes describing different aspects of its behavior.

We chose ten electromechanical devices from several encyclopedias [2, 25, 38]. We carefully selected these devices to have similar components that needed different models. The devices range in complexity from 10 to 54 components. The bimetallic strip temperature gauge in Figure 1 is one of the devices. The most complex one, a car distributor system, is shown in Figure 10. The function of the distributor is to ensure that the spark plugs in the piston chamber fire in sequence at the right time. It works as follows: as the cam rotates, it opens the contact breaker, causing the current in the primary windings to drop rapidly (the condenser prevents a spark

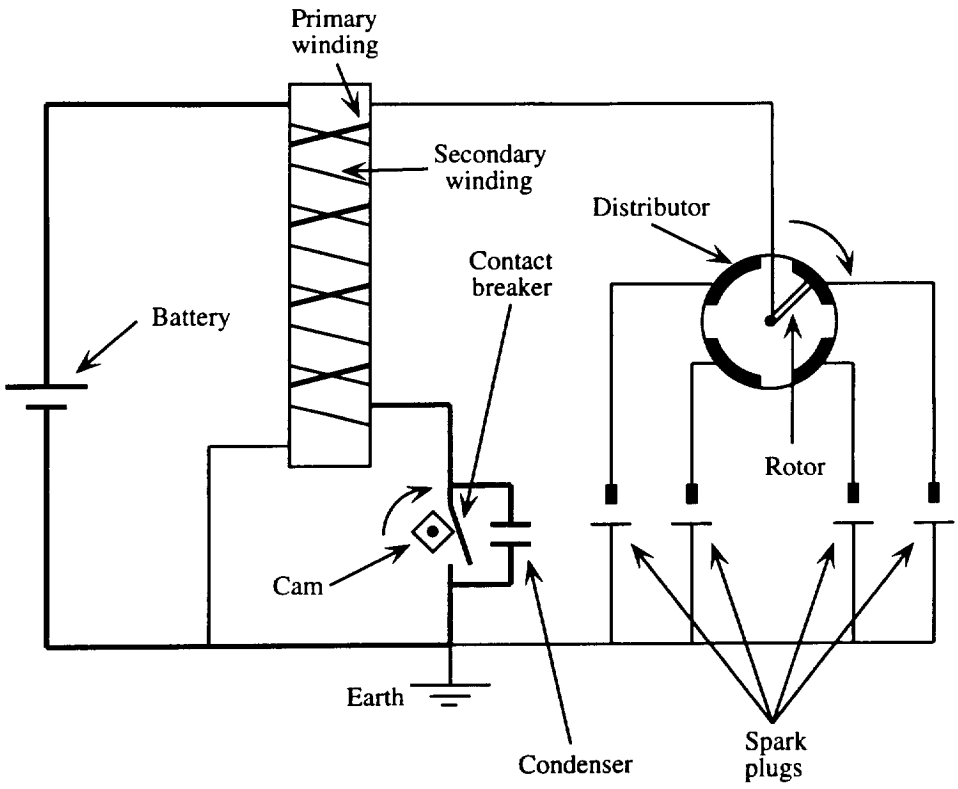

Figure 10: Car distributor system

from jumping across the contact breaker). The rapid change in current in the primary winding causes a large induced electromotive force in the secondary winding. At the same time, the distributor rotor connects the secondary winding to one of the spark plugs (the rightmost spark plug in the figure). The large induced electromotive force causes a spark to jump across the spark plug. Descriptions of the other devices can be found in [28].

Table 3 summarizes the results of our experiments. The first and second column show the device name and the number of components in each device. This number includes the components created by the program to account for possible component interaction paths. The third column shows the number of operating regions of each device that the model selection program was run on. The fourth column shows the estimated number of consistent and complete device models. These models have at most one model fragment from each assumption class, and a model fragment from each required assumption class. The estimates, derived from the size and organization of the knowledge base, clearly show that a brute-force search for adequate models is totally impractical. The fifth column shows the total number of models actually examined by the program. This is the sum of the number of models examined when the program constructs an initial model and then simplifies it. These low numbers show why our model selection method is practical. The last column shows the actual

Device name	# of components	# of regions	Estimated space	Generated space	Time (sec)
Bimetallic strip temperature gauge	12	1	3.8e16	46	28.4
Bimetallic strip thermostat	10	2	5.4e12	85	43.7
Flexible wire temperature gauge	13	1	2.6e20	78	59.9
Galvanometer temperature gauge	19	1	6.1e31	120	149.8
Electric bell	22	2	6.6e40	117	262.4
Magnetic sizing device	22	1	2.1e51	117	456.0
Carbon pile regulator	26	1	1.5e49	115	262.5
Electromagnetic relay thermostat	30	3	8.7e49	293	472.7
Tachometer	34	1	6.8e58	195	503.6
Car distributor system	54	1	9.9e72	160	352.6

Table 3: Summary of experimental results

run time on a Texas Instruments Explorer II workstation.[4]

Table 4 shows the characteristics of the models. The first three columns show the number of model fragments in the most accurate model, in the initial model, and in the (final) adequate model for each device. Multiple entries for a single device correspond to running the program on the different operating regions of that device. The last two columns show the number of equations in the initial and adequate models. Automatically generated device models usually contain a very large number of equations and parameters (e.g., a current for each end of an electrical conductor, and the corresponding Kirchhoff's Current Law equations). We drastically decrease the number of equations by elementary simplifications, e.g., replacing equals by equals. The number of equations reported in Table 4 is the number remaining after such simplifications.

Note that the number of model fragments in the initial model is about half the number in the most accurate model. This demonstrates that the heuristic method is effective in finding an initial model that is significantly simpler than the most accurate model. Note also that, in most cases, the adequate model has on average two-thirds less model fragments than the initial model. This indicates that the heuristic method of finding an initial model is not, by itself, sufficient to find an adequate model, or even a model that is close to being an adequate model; it must be simplified as described in the previous section.

In conclusion, these results show that the space of complete and consistent device models is too large for brute-force search. The use of causal approximations, as implemented in the model composition algorithm described above, allows the program to systematically explore only a tiny fraction of the search space, making model

[4]Significantly faster runs have been observed on different machines using different Lisp implementations. For example, the tachometer example has been run in a little over a minute and a half on a Sparc Station 2 under Lucid Lisp version 4.1 [Jon L White, personal communication].

Device name	Number of model fragments			Number of model equations	
	Most accurate model	Initial model	Adequate model	Initial	Adequate
Bimetallic strip temperature gauge	75	36	27	33	13
Bimetallic strip thermostat	54	38	14	31	6
	54	39	31	36	11
Flexible link temperature gauge	94	60	25	52	13
Galvanometer temperature gauge	154	98	28	93	17
Electric bell	177	7	6	6	6
	177	108	45	122	35
Magnetic sizing device	202	122	43	126	32
Carbon pile regulator	211	122	51	131	36
Electromagnetic relay thermostat	211	117	31	102	13
	211	119	36	119	30
	211	74	14	73	3
Tachometer	285	170	44	164	30
Car distributor system	348	178	28	192	21

Table 4: Characteristics of the models.

selection practical. We also conclude that the heuristic technique for finding an initial model is effective in finding significantly simpler than the most accurate model, but is insufficient for finding adequate models. It still needs to be significantly simplified to produce an adequate model.

7 Related work

Our work shares the motivation and builds upon Falkenhainer and Forbus' work on compositional modeling [12, 13]. Compositional modeling creates models by composing model fragments selected from a model fragment library. Each model fragment is conditioned on a set of *modeling assumptions* which explicate the approximations, perspectives, granularity, and operating assumptions underlying the model fragment. Mutually contradictory assumptions are organized into assumption classes, and a set of domain-independent and domain-dependent constraints are used to govern the use of modeling assumptions. An adequate device model contains all the terms mentioned in a user query, and uses only model fragments that are entailed by a set of mutually consistent assumptions that satisfy all the constraints. Adequate models are constructed by using a variant of constraint satisfaction called *dynamic constraint satisfaction* [26], and then validated using either qualitative or numerical simulation. If the validation discovers any inconsistencies, the process is repeated with this additional information.

Our work explicitly focuses on compositional modeling for generating causal ex-

planations from structural and functional descriptions of devices. As argued in the introduction, causal explanations are essential for tutoring systems and play a key role in the analysis, design, and diagnosis of devices.

Models that support causal explanations are necessary for generating causal explanations and are useful for answering queries, running simulations, and making predictions. They can be used as base models that can be fine-tuned and incrementally modified for other tasks. The focus on causal explanations allowed us to develop new techniques and improve key aspects of compositional modeling. Specifically,

- we improved the knowledge representation by refining and augmenting the relations between model fragments in the library and organizing them into the generalization, possible models, and approximation hierarchies. The organization provides compact representations, facilitates knowledge-base construction and maintenance, and supports efficient model fragment retrieval and device model generation.

- we introduced causal approximations, which are commonplace in physics and engineering, demonstrated their use in model fragment selection, and built a usable medium-sized model fragment library in which all approximations are causal approximations.

- we redefined model adequacy with respect to generating causal explanations of the expected behavior. The expected behavior provides more constraints on model adequacy than a query since not only does it specify the terms that must be included in an adequate model, but it also specifies the causal relations required between parameters. This additional constraint leads to the expected behavior being the central determinant of model adequacy, thereby diminishing the importance of the modeling constraints. Hence, we are able to use less expressive constraints, avoiding the intractability inherent in constraint satisfaction.

- we developed an effective order-of-magnitude reasoning technique for behavior generation as an alternative to qualitative and numerical techniques. Order-of-magnitude reasoning is at the appropriate level of abstraction: is better able to discriminate between models than qualitative techniques, and is more efficient and robust than numerical techniques.

- we showed that compositional modeling for causal explanation generation is tractable when all approximation relations between model fragments are causal approximations. We replaced the ATMS-based dynamic constraint satisfaction technique by an efficient model composition algorithm that exploits the properties of causal approximations.

- we avoid instantiating the complete domain theory for the scenario description. Only those model fragments that are actually used in the search process are instantiated.

- we model devices with multiple operating regions differently: they handle multiple operating regions by selecting a single model for all the regions, although this model may not be appropriate for all the operating regions. We handle only one region at a time, so different models can be chosen for different regions. This requires a new set of inputs for each operating region.

The result is an efficient model composition algorithm for generating causal approximations and the models supporting them. Finally, note that the original compositional modeling algorithm [13] cannot be easily adapted to the task of generating causal approximations for expected behaviors. In that algorithm, the constraints play a central role in defining model adequacy, and any task focus has to be embedded in these constraints. Embedding such a task focus is, in general, not easy; in particular, it is not clear how the expected behavior can be expressed as a set of declarative constraints. Furthermore, restricting the expressivity of these constraints is not always possible, so that they require potentially exponential-time constraint satisfaction during model selection.

Nayak introduced and rigorously analyzed the theoretical basis of causal approximations in [29, 31]. The model composition algorithm developed there simplifies the most accurate model. We built upon this work and extended it in four ways. First, we showed how, in practice, class level descriptions of model fragments should be represented and organized within a model fragment library, including the use of the possible models and generalization hierarchies. Second, we extended the model selection algorithm by including behavior generation using order of magnitude reasoning. This allowed us to ensure that all significant phenomena are included in adequate models. Third, we introduced the component interaction heuristic, which allowed us to find an initial model that is significantly simpler than the most accurate model. Fourth, we tested our implementation on a variety of examples, providing empirical validation of the theoretical claims of [29, 31].

The work on *Graphs of Models* [1] discusses a technique for selecting models of acceptable accuracy. A graph of models is a graph in which the nodes are models and the edges are assumptions that have to be changed in moving from one model to another. A model in this graph has acceptable accuracy if its predictions are free of conflicts, which are detected either empirically or internally. Empirical conflicts are detected by experimentally verifying a model's predictions, while internal conflicts are detected by checking the model's predictions against a set of consistency rules that capture the model's assumptions. When a conflict is detected, a set of domain-dependent *parameter change* rules help to select a more accurate model, and the above process is repeated. Analysis begins with the simplest model in the graph of models, and terminates when an accurate enough model has been found.

The main difference between the Graphs of Models approach and compositional modeling is that the Graphs of Models approach explicitly represents models, while compositional modeling implicitly represents models as a set of model fragments that can be combined to produce device models. Compositional modeling leads to greater

flexibility in tailoring models to specific situations and can potentially represent exponentially many models in the number of model fragments. To get comparable flexibility in the Graphs of Models approach requires an explicit representation of an exponentially large space of device models, which is clearly impractical. An advantage of the Graphs of Models approach is that it identifies well-understood models and can associate with them more efficient specialized problem solvers instead of a general purpose problem solver that is applicable to all models.

The consistency rules used to verify a model's predictions are similar to our behavioral preconditions and coherence constraints. However, we do not validate a model's predictions empirically, and we have not explicitly addressed the problem of switching to a more accurate model in light of an empirical conflict. Our techniques are best viewed as providing an intelligent method for selecting an initial model. Since they always start the analysis with the simplest model, making no effort to identify a better starting model, our techniques are complementary to theirs: select an initial model using our technique, and do model switching using theirs.

Using approximations to guide modeling has been recently investigated by Weld [42]. This work introduces an interesting class of approximations called *fitting approximations*. Informally, a model M_2 is a fitting approximation of a model M_1 if M_1 contains an exogenous parameter, called a fitting parameter, such that the predictions using M_1 approach the predictions using M_2, as the fitting parameter approaches a limit. Weld shows that when all the approximations are fitting approximations, the domain-dependent parameter change rules discussed above can be replaced by a domain-independent technique for model switching. Fitting approximations and causal approximations are fundamentally incomparable because the former talks about behavior differences, while the latter talks about causal dependencies. However, in practice, it appears that fitting approximations are also causal approximations, making Weld's domain-independent technique for model switching amenable for use in our system.

Williams' work on producing *critical abstractions* [48] shares our motivations for finding adequate models—we are both striving to find parsimonious descriptions of how a device works. A critical abstraction is a parsimonious description of a device relative to a set of questions. Given a device model, he constructs a critical abstraction in three steps: (a) eliminating superfluous interactions; (b) aggregating interactions that are local to a single mechanism using symbolic algebra; and (c) further abstracting the aggregated interactions. Williams' abstraction process is similar to our model simplification procedure. Specifically, his first step, which eliminates superfluous interactions, is similar to our simplification operator that drops irrelevant model fragments. The primary difference between our approaches is one of emphasis: we have focussed on the problem of selecting approximations from a prespecified space of possible approximations, while he has focussed on finding techniques for automatically abstracting a base model.

Davis's work on model-based diagnosis [7] has been one of the original inspirations for our work. Davis describes a diagnostic method based on tracing paths of causal

interactions. He argues that the power of the approach stems not from the specific diagnostic method, but from the model which specifies the allowed paths of causal interaction. He shows that efficient diagnosis, while retaining completeness, can be obtained by initially considering models with only a few paths of interactions, and adding in additional paths when the model fails to account for the symptoms. Our simplicity ordering on models follows Davis' diagnostic technique: diagnosis starts at an adequate model, with successively more complex models being used if a model is unable to account for the symptoms. The use of causal approximations ensures that using more complex models will add new paths of causal interaction. We have used Davis' definition of component adjacency (two components are adjacent if they can interact with each other by some means). In particular, the component interaction heuristic is closely related to the notion of adjacency: adjacent components must have compatible models.

Reasoning about accuracy is a key aspect of generating adequate device models: a model must be sufficiently accurate to be useful. Recent work has investigated the issues involved in selecting models of acceptable accuracy [10, 11, 27, 33, 36, 43]. In this paper we have not developed sophisticated techniques for reasoning about model accuracy. A model is deemed to be accurate enough if it satisfies all the behavioral preconditions and coherence constraints, with different levels of accuracy corresponding to different settings of the thresholds. However, our system does not reason about the settings of the thresholds, whose values are part of the input.

In other related work, Liu and Farley present a query-driven method for selecting and shifting between macroscopic and microscopic domain theories [24]. The selection and shift of ontologies is driven by a set of *ontological choice rules*. Iwasaki and Levy show how relevance reasoning can be used for efficiently selecting model fragments for simulation [20]. The efficiency of their algorithm is also based on the use of causal approximations. The primary difference is that they replace our component interaction heuristic with backward chaining along causal influences. Schut and Bredeweg show how the parsimony of a model composed of model fragments can be further enhanced by removing irrelevant model particles, which include system elements, parameters, parameter values, and parameter relations [34].

8 Conclusion and Future Work

Causal explanations, and the models that support them, play a central role in many reasoning tasks, such as analysis, design, and diagnosis. This paper presents an implemented polynomial-time model composition algorithm for constructing adequate models that provide parsimonious causal explanations of the functioning of a device. The algorithm is based on compositional modeling, a modeling framework in which adequate device models are constructed by composing model fragments selected from a model fragment library. The focus on causal explanations allowed us to develop new techniques for compositional modeling and improve key aspects of the model

composition process. To model important aspects of device function, we introduce expected behaviors, an abstract, causal accounts of *what* a device does. To focus the search for a model and guarantee efficient model construction, we use a new approximation relationship between model fragments, called a causal approximation. For efficient model fragment retrieval and model generation, we organize model fragments into various hierarchies. For efficient model validation, we use causal ordering and a new logarithm-based order of magnitude reasoning technique. We have implemented the model composition algorithm and produced adequate models and causal explanations for a variety of electromechanical devices selected from several engineering encyclopedias.

We believe that the task of explaining expected behaviors is important in its own right and complements the task of answering user queries, as addressed in [13]. Expected behaviors typically define the context in which queries will be asked. Successive queries are generally related to each other; constructing a device model from scratch for each query is impractical and inefficient, especially for complex devices. A better strategy, widely used in engineering, is to construct first a base device model suitable for a class of queries and then modify, refine, or specialize this device model for each query as necessary. Furthermore, expected behaviors contain more information than queries because they not only specify the terms and objects that must be included in the model, they also also specify their causal relationships. This leads to better base models and more focused model construction. Expected behaviors provide a natural way of describing the key elements, properties, and relations that apply to a class of queries. Models that support causal explanations of expected behavior are more durable and easier to update than models specifically tailored to a particular query.

The model composition algorithm presented in this paper can be easily adapted for incremental model revision and modification. Incremental model revision is necessary to answer a series of queries, to account for more causal relationships, or to refine an existing model. Instead of starting the algorithm with an empty model, model composition starts with a base model that is validated and modified as described in steps 1–6 of the model composition algorithm in Figure 9. Incremental model modification avoids having to construct a new model from scratch in each situation.

The work described in this paper can be extended in a number of different ways. We briefly discuss some next.

The expected behavior, captured as a causal relation between parameters, has proved to be a useful characterization of device function. It is both expressive and can be checked efficiently using causal ordering techniques. Clearly, more expressive languages will allow us to represent a wider range of expected behaviors. For example, we may want specify the relative directions of parameter change (increasing T_t causes θ_p to increase), the functional relationships between parameters (T_t and θ_p are linearly related), or the different aspects of the device's *function*, as discussed in the functional reasoning literature [5]). Developing more expressive languages is in itself not difficult; the real challenge is to develop more expressive *tractable* languages. Intractable languages compromise the effectiveness of selecting adequate device models,

and hence of problem solving. An important research direction is thus the development of more expressive languages that allow efficient model composition algorithms.

Accuracy is a very important characteristic of adequate device models. As discussed earlier, we have not developed sophisticated methods for reasoning about model accuracy. An important direction for future work is to allow the user to specify the desired accuracy of the model easily, e.g., by specifying tolerances on certain parameters. We then need to develop techniques for finding models that guarantee that predictions will lie within the specified tolerances.

Many devices go through multiple operating regions during the course of their normal operations. Different operating regions can have different characteristics, requiring the use of different models. We are currently investigating how to generalize the single-region model composition algorithm to handle multiple operating regions. This will require (a) generalizing the present order of magnitude reasoning technique to allow temporal simulation; and (b) techniques for inferring the expected behavior of each operating region, given the overall expected behavior of the device.

We believe that the compositional modeling framework can be effectively applied to a variety of tasks. Most prominently, we see its use in producing adequate device models for fault diagnosis and monitoring, where recent research has shown an emerging understanding of model adequacy [7, 17]. We believe that the techniques developed in this paper will prove valuable in developing methods for building adequate models for diagnosis and monitoring.

Acknowledgements

We would like to thank Edward Feigenbaum, Richard Fikes, Brian Falkenhainer, Renate Fruchter, Andy Golding, Nita Goyal, Yumi Iwasaki, Rich Keller, Alon Levy, Rajan Ramaswamy, Rich Washington, Dan Weld, and Michael Wolverton for useful discussions and for comments on earlier drafts. Sanjaya Addanki collaborated in the initial phase of this project. Pandurang Nayak was supported by an IBM Graduate Technical Fellowship. Additional support for this research was provided by the Defense Advanced Research Projects Agency under NASA Grant NAG 2-581 (under ARPA order number 6822), by NASA under NASA Grant NCC 2-537, and by IBM under agreement number 14780042.

References

[1] Sanjaya Addanki, Roberto Cremonini, and J. Scott Penberthy. Graphs of models. *Artificial Intelligence*, 51:145–177, 1991.

[2] Ivan I. Artobolevsky. *Mechanisms in Modern Engineering Design*, volume 5. Mir Publishers, Moscow, 1980.

[3] D. Bobrow, editor. *Qualitative Reasoning About Physical Systems.* North-Holland, 1984.

[4] John Seely Brown, R. R. Burton, and Johan de Kleer. Pedagogical, natural language and knowledge engineering techniques in SOPHIE I, II and II. In Derek Sleeman and John Seely Brown, editors, *Intelligent Tutoring Systems*, pages 227–282. Academic Press, New York, 1982.

[5] B. Chandrasekaran, editor. *IEEE Expert* 6(2). April 1991.

[6] James Crawford, Adam Farquhar, and Benjamin Kuipers. QPC: A compiler from physical models into qualitative differential equations. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 365–372. American Association for Artificial Intelligence, 1990.

[7] Randall Davis. Diagnostic reasoning based on structure and behavior. *Artificial Intelligence*, 24:347–410, 1984.

[8] Johan de Kleer. Multiple representations of knowledge in a mechanics problem-solver. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 299–304. International Joint Conferences on Artificial Intelligence, Inc., 1977.

[9] Johan de Kleer and John Seely Brown. A qualitative physics based on confluences. *Artificial Intelligence*, 24:7–83, 1984.

[10] Thomas Ellman, John Keane, and Mark Schwabacher. Intelligent model selection for hillclimbing search in computer-aided design. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 594–599. American Association for Artificial Intelligence, AAAI Press/The MIT Press, July 1993.

[11] Brian Falkenhainer. Ideal physical systems. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 600–605. American Association for Artificial Intelligence, AAAI Press/The MIT Press, July 1993.

[12] Brian Falkenhainer and Kenneth D. Forbus. Setting up large-scale qualitative models. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 301–306. American Association for Artificial Intelligence, 1988.

[13] Brian Falkenhainer and Kenneth D. Forbus. Compositional modeling: Finding the right model for the job. *Artificial Intelligence*, 51:95–143, 1991.

[14] Kenneth D. Forbus. Qualitative process theory. *Artificial Intelligence*, 24:85–168, 1984.

[15] Kenneth D. Forbus. The qualitative process engine. In Daniel S. Weld and Johan de Kleer, editors, *Readings in Qualitative Reasoning about Physical Systems*, pages 220–235. Morgan Kaufmann, 1990.

[16] Kenneth D. Forbus and A. Stevens. Using qualitative simulation to generate explanations. In *Proceedings of the Third Annual Meeting of the Cognitive Science Society*, pages 219–221, 1981.

[17] Walter C. Hamscher. Modeling digital circuits for troubleshooting. *Artificial Intelligence*, 51:223–271, 1991.

[18] Jerry R. Hobbs. Granularity. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 432–435. International Joint Conferences on Artificial Intelligence, Inc., 1985.

[19] Yumi Iwasaki. Causal ordering in a mixed structure. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 313–318. American Association for Artificial Intelligence, August 1988.

[20] Yumi Iwasaki and Alon Y. Levy. Automated model selection for simulation. In *Working Papers of the Seventh International Workshop on Qualitative Reasoning about Physical Systems*, 1993.

[21] Yumi Iwasaki and Chee-Meng Low. Model generation and simulation of device behavior with continuous and discrete changes. Technical Report KSL 91-69, Stanford University, Knowledge Systems Laboratory, 1991.

[22] Yumi Iwasaki and Herbert A. Simon. Causality in device behavior. *Artificial Intelligence*, 29:3–32, 1986.

[23] Benjamin Kuipers. Qualitative simulation. *Artificial Intelligence*, 29:289–338, 1986.

[24] Zheng-Yang Liu and Arthur M. Farley. Shifting ontological perspective in reasoning about physical systems. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 395–400. American Association for Artificial Intelligence, AAAI Press/MIT Press, July 1990.

[25] David Macaulay. *The Way Things Work.* Houghton Mifflin Company, Boston, 1988.

[26] Sanjay Mittal and Brian Falkenhainer. Dynamic constraint satisfaction. In *Proceedings Eighth National Conference on Artificial Intelligence*, pages 25–32. American Association for Artificial Intelligence, AAAI Press/MIT Press, July 1990.

[27] P. Pandurang Nayak. Validating approximate equilibrium models. In *Proceedings of the 1991 Model-Based Reasoning Workshop*, July 1991.

[28] P. Pandurang Nayak. *Automated Modeling of Physical Systems.* PhD thesis, Stanford University, Department of Computer Science, Stanford, CA, 1992.

[29] P. Pandurang Nayak. Causal approximations. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 703–709. American Association for Artificial Intelligence, July 1992.

[30] P. Pandurang Nayak. Order of magnitude reasoning using logarithms. In B. Nebel, C. Rich, and W. Swartout, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference (KR92)*, San Mateo, CA, 1992. Morgan Kaufmann.

[31] P. Pandurang Nayak. Causal approximations. 1993. Submitted for publication.

[32] Ramesh S. Patil, Peter Szolovits, and William B. Schwartz. Causal understanding of patient illness in medical diagnosis. In *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, pages 893–899. International Joint Conferences on Artificial Intelligence, Inc., 1981.

[33] Olivier Raiman and Brian C. Williams. Caricatures: Generating models of dominant behavior. In *Proceedings of the AAAI-92 Workshop on Approximation and Abstraction of Computational Theories*, 1992.

[34] Cis Schut and Bert Bredeweg. Automatic enhancement of model parsimony. In *Working Papers of the Seventh International Workshop on Qualitative Reasoning about Physical Systems*, 1993.

[35] D. Serrano and David C. Gossard. Constraint management in conceptual design. In D. Sriram and R. A. Adey, editors, *Knowledge Based Expert Systems in Engineering: Planning and Design*, pages 211–224. Computational Mechanics Publications, 1987.

[36] Mark Shirley and Brian Falkenhainer. Explicit reasoning about accuracy for approximating physical systems. In *Working Notes of the Automatic Generation of Approximations and Abstractions Workshop*, pages 153–162, July 1990.

[37] Herbert A. Simon. On the definition of the causal relation. *Journal of Philosophy*, 49:517–528, 1952.

[38] C. van Amerongen. *The Way Things Work.* Simon and Schuster, 1967.

[39] J. W. Wallis and E. H. Shortliffe. Explanatory power for medical expert systems: Studies in the representation of causal relationships for clinical consultations. *Methods Inform. Med.*, 21:127–136, 1982.

[40] Sholom M. Weiss, Casimir A. Kulikowski, Saul Amarel, and Aran Safir. A model-based method for computer-aided medical decision-making. *Artificial Intelligence*, 11:145–172, 1978.

[41] Daniel S. Weld. Explaining complex engineered devices. Technical Report TR-5511, BBN, Cambridge, MA, 1983.

[42] Daniel S. Weld. Approximation reformulations. In *Proceedings Eighth National Conference on Artificial Intelligence*, pages 407–412. American Association for Artificial Intelligence, AAAI Press/MIT Press, July 1990.

[43] Daniel S. Weld and Sanjaya Addanki. Query-directed approximation. In Boi Faltings and Peter Struss, editors, *Recent Advances in Qualitative Physics.* MIT Press, Cambridge, MA, 1991.

[44] Daniel S. Weld and Johan de Kleer, editors. *Readings in Qualitative Reasoning About Physical Systems.* Morgan Kaufmann Publishers, Inc., San Mateo, California, 1990.

[45] Brian C. Williams. Qualitative analysis of MOS circuits. *Artificial Intelligence*, 24:281–346, 1984.

[46] Brian C. Williams. *Invention from First Principles via Topologies of Interactions.* PhD thesis, M.I.T., 1989.

[47] Brian C. Williams. Interaction-based invention: Designing novel devices from first principles. In *Proceedings Eighth National Conference on Artificial Intelligence*, pages 349–356. American Association for Artificial Intelligence, AAAI Press/MIT Press, July 1990.

[48] Brian C. Williams. Critical abstraction: Generating simplest models for causal explanation. In *Proceedings of the Fifth International Workshop on Qualitative Reasoning about Physical Systems*, May 1991.